# System Programming Assignment

Sharvansh Shukla 2019CSC1016

## The problem:

Consider a simple SQL query that retrieves a character string from a column of the Table “formulae” as below:

SELECT formula FROM formulae WHERE id = 5

The returned value may be like:

‘(<A1>+<A2>)*0.5 - <A3>’

where <A1>, <A2>, <A3> etc. Are to be substituted by the fieldnames stored in a Table

“formula_fields”. An example schema can be:

(“id”, “varname”,”colname”)

(1,’<A1>’,’Salary.Basic’)

(2,’<A2>’,’Salary.TA’)

(3,’<A3>’,’Salary.PF’)

(i.e, tablename.columnname is stored in the third column of the table)

Having obtained the formula from the table by a query such as above, you need to convert it to a corresponding SQL query. For the example above, the resulting query should be

SELECT ((Basic + TA) * 0.5 – PF) FROM salary WHERE ... (some condition here)

To do:

1. Use lex (flex) to scan and interpret formula that is always constructed from symbols of the form <A1>, <A2> etc., and numeric literals, and arithmetic operators +, -, *, / and () [2 Marks]

2. Use yacc (bison) to generate parser for checking if formula is correctly formed according to standard grammar of arithmetic expressions involving tokens from 1 above [3 Marks]

3. Generate code equivalent to the example SQL code given above by specifying appropriate instructions in parser using C/C++. [2 Marks]

4. Make the tables “Formulae” and “Salary” in some DBMS, say MySQL, and test run the successful execution of the generated query. [2 Marks}

5. Include documentation for your design, standard comments in source code, and any special assumptions you make. [1 Mark]

## The solution:

Parts 1 through 4 have been answered via means of three files that are appended to this pdf, namely hsr.y (bison), hsr.l (flex), and sys_prog.db (database).

As for Part 5, well, this pdf makes up for a big chunk of the aforementioned documentation, along side the comments inside the source code.

## SQLite3

One important thing to note is that you will need SQLite3 to run the DBMS queries in this question, so in case you haven’ t installed it, I have also attached the required files needed to run SQLite3 on your machine.

All you need to do is paste the files where you’re running the program and type the following command in the terminal to create an object file for SQLite3:

*gcc sqlite3.c -c*

## Executing the code:

The code can be simply executed using the following commands in the terminal:

*bison -d hsr.y*

*flex hsr.l*

*g++ -o solution lex.yy.c hsr.tab.c sqlite3.o -Ipath\to\sqlite3*

And you can run the solution through this command:

*solution.exe*

or if you’re more of a Linux person, use:

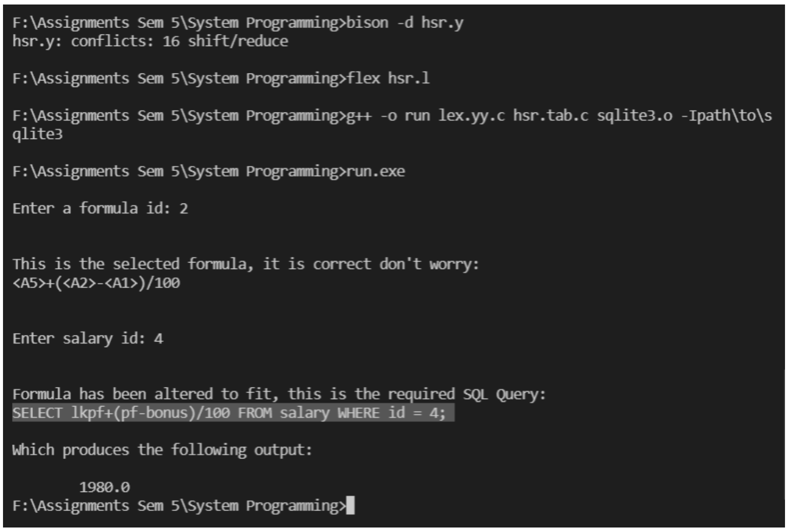
*./solution*

## Conflicts

I have encountered several shift/reduce conflicts in my source while making the CFG for this solution, but have b=not been successful to remove them, so that might prove to be a lose end on an evaluation scale. The good news, however, is that it does not affect the working of the code at all, so we are golden when it come to that.

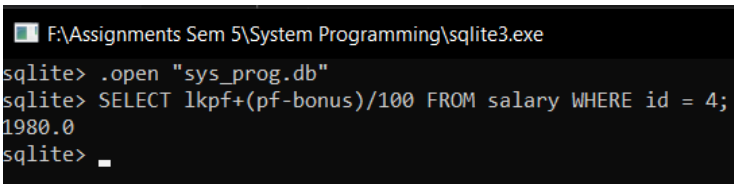
## Outputs:

The following screenshots are a representation of the kind of output you can expect when you run the code in your terminal:



```
F:\Assignments Sem 5\System Programming>bison -d hsr.y
hsr.y: conflicts: 16 shift/reduce

F:\Assignments Sem 5\System Programming>flex hsr.l

F:\Assignments Sem 5\System Programming>g++ -o run lex.yy.c hsr.tab.c sqlite3.o -Ipath\to\s
qlite3

F:\Assignments Sem 5\System Programming>run.exe

Enter a formula id: 2


This is the selected formula, it is correct don't worry:
<A5>+(<A2>-<A1>)/100


Enter salary id: 4


Formula has been altered to fit, this is the required SQL Query:
SELECT lkpf+(pf-bonus)/100 FROM salary WHERE id = 4;

Which produces the following output:

        1980.0
F:\Assignments Sem 5\System Programming>
```

Also, here is the same query running in an SQLite3 environment, independently and fetching the same output, henceforth verifying the correctness of the solution:



```
sqlite> .open "sys_prog.db"
sqlite> SELECT lkpf+(pf-bonus)/100 FROM salary WHERE id = 4;
1980.0
sqlite> _
```

## Conclusion:

Well, there you have it. A flex/bison program that reads a formula, parses it to check its validity, replaces the placeholders with actual formulating terms, and calls an SQL query to fetch output.

Here's hoping this assignment meets the required standards for evaluation.